# Parallel Corpus Creation

Pavan Mandava, Isaac Riley, Jasvinder Singh

January 31, 2020

# Overview

1. Motivation
2. Project Workflow
   - Collect
   - Prepare
   - Access
3. Potential Enhancements / Applications

# What is a parallel corpus?

A corpus containing the same text in two or more languages - especially valuable for machine translation, more useful when aligned.

# Uses of parallel corpora

- machine translation
- translation analysis & studies
- comparative linguistics
- digital humanities
- technology-assisted language learning

# Examples of Parallel Corpora

- Rosetta Stone
- OPUS (collection - incudes OpenSubtitles, TED Talks, etc.)
- EuroParl Corpus
- EUR-Lex Corpus
- Tatoeba

# Our Contribution

We create a robust toolkit for creating and querying parallel corpora, with support for XML, HTML, and SQL.

# Project Workflow

# Collect

1. ~~WikiSource API~~
2. ~~Download epub and convert to .txt~~
3. Easy way out: XML-like .fb2 files
    - each book has its own structure conforming to a common XSD
    - used XPath and XSD to obtain structure

→ Not too much automatic collection. A more advanced implementation could make use of crawling, but we gathered data by hand.

# Prepare

1. find texts
2. segment texts into chapters
3. align chapters at sentence level
4. write to XML files
5. create SQL database from XML files

# Bitext alignment in general

Problem: Given two parallel texts, identify corresponding units (usually chapters, paragraphs, sentences, potentially even words)

# Alignment algorithm

1. clean texts (esp. ellipses with spaces)
2. tokenize sentences (nltk.sent_tokenize) and create dataframes (Pandas)
3. translate sentences (Google NMT API)
4. calculate relative position based on cumulative amount of text each sentence represents
5. find “anchor matches” - all high-quality matches within window $\omega$ of overlapping sentences
6. align iteratively between anchors by comparing next sentence $s_i^{lang_1}$ with $concat(s_j^{lang_2}, ..., s_{j+\mu}^{lang_2})$ and vice versa, where $\mu$ is our lookahead (typically 4 works well)

# Very Simple Example

Alice fing an sich zu langweilen; sie saß schon lange bei ihrer Schwester am Ufer und hatte nichts zu thun. Das Buch, das ihre Schwester las, gefiel ihr nicht; denn es waren weder Bilder noch Gespräche darin. „Und was nützen Bücher," dachte Alice, „ohne Bilder und Gespräche?"

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

# Very Simple Example: Sentence Tokenization

1. Alice fing an sich zu langweilen; sie saß schon lange bei ihrer Schwester am Ufer und hatte nichts zu thun.
2. Das Buch, das ihre Schwester las, gefiel ihr nicht; denn es waren weder Bilder noch Gespräche darin.
3. „Und was nützen Bücher,“ dachte Alice, „ohne Bilder und Gespräche?“

1. Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, ‘and what is the use of a book,’ thought Alice ‘without pictures or conversation?’

# Very Simple Example: Translate Sentences (DE → EN)

1. Alice started to get bored; she had been sitting with her sister on the bank for a long time and had nothing to do.
2. She didn’t like the book her sister read; because there were no pictures or conversations in it.
3. „And what use are books,“ thought Alice, „without pictures and conversations?“

1. Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, ‘and what is the use of a book,’ thought Alice ‘without pictures or conversation?’

# Very Simple Example: Alignment Step

$trans\_sim(a, b) = 1 - levenshtein(a, b)/max(length(a), length(b))$
$trans\_sim(a1, b1) = 0.272$
$trans\_sim(a1 + a2, b1) = 0.395$
$trans\_sim(a1 + a2 + a3, b1) = 0.591 > trans\_sim(a1 + a2 + a3 + a4, b1)$
$\Rightarrow$ new alignment (indices inclusive): $((1, 3), (1, 1))$

# XML Structure

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<book code="cap_book">
  <bookInfo>
    <title>Crime and Punishment</title>
    <lang>en</lang>
    <isTranslation>true</isTranslation>
    <totalChapters>2</totalChapters>
    <source>https://en.wikisource.org/wiki/</source>
    <description> <!--Optional-->
      Crime and Punishment (Russian: Преступление и наказание)
      is a novel written by Russian author Fyodor Dostoevsky.
    </description>
    <isbn>n.a.</isbn> <!--Optional-->
    <author>Fyodor Dostoevsky</author>
    <author translator="true">Constance Garnett</author>
  </bookInfo>
  <content>
    <chapter num="1" name="Chapter 1">
      <sentence num="1">First Sentence</sentence>
      <sentence num="2">Second Sentence</sentence>
      <sentence num="3">Third Sentence</sentence>
    </chapter>
    <chapter num="2" name="Chapter 2">
      <sentence num="1">First Sentence</sentence>
      <sentence num="2">Second Sentence</sentence>
      <sentence num="3">Third Sentence</sentence>
    </chapter>
  </content>
</book>
```

# Saving to XML

- Take output from aligner and save to XML file
- Add entry to JSON file to record file path

```
“book_code": [
    {
        "xml_file": “<file-name>",
        "lang": "en",
        "xml_file_path": “<file-path>",
        "is_validated": false,
        "is_saved_to_db": false
    }
]
```

# XSLT

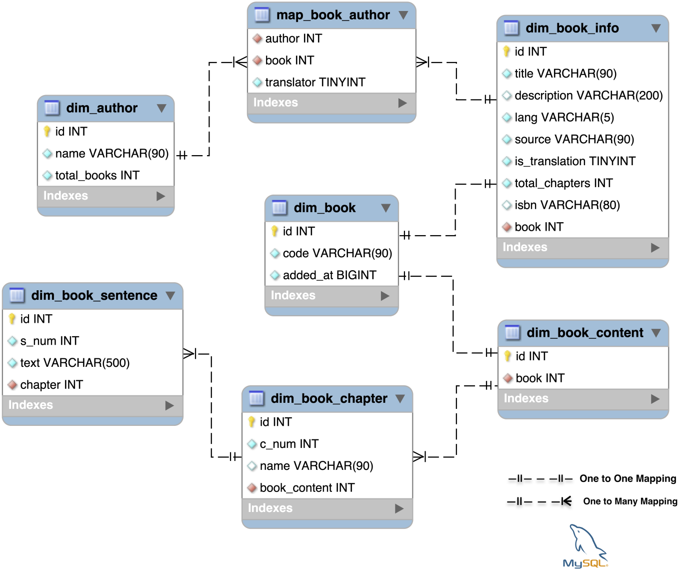
Generated html using XSLT, available online for all bi- and trilingual Dostoevsky enthusiasts:

Human-readable bitexts

# Need for Database

Dilemma: The XML/HTML formats are good for reading, less ideal for queries.
Solution: SQL database (MySQL)

# Structure of DB Schema



**map_book_author**
- author INT
- book INT
- translator TINYINT
- Indexes

**dim_book_info**
- id INT
- title VARCHAR(90)
- description VARCHAR(200)
- lang VARCHAR(5)
- source VARCHAR(90)
- is_translation TINYINT
- total_chapters INT
- isbn VARCHAR(80)
- book INT
- Indexes

**dim_author**
- id INT
- name VARCHAR(90)
- total_books INT
- Indexes

**dim_book**
- id INT
- code VARCHAR(90)
- added_at BIGINT
- Indexes

**dim_book_sentence**
- id INT
- s_num INT
- text VARCHAR(500)
- chapter INT
- Indexes

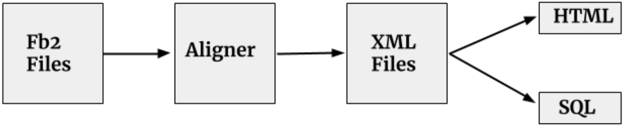
**dim_book_content**
- id INT
- book INT
- Indexes

**dim_book_chapter**
- id INT
- c_num INT
- name VARCHAR(90)
- book_content INT
- Indexes

–||– – –||– One to One Mapping

–||– – –|< One to Many Mapping

# Project Overview



# Potential Enhancements / Applications

Enhancements:

- Expand corpus
- Alignment at word level
- Local (and pre-trained ) NMT engine instead of querying online server
- Robust chapter segmentation and pre-processing to take noisier and less-structured text formats

Applications:

- Compare different translation styles
- Investigate translation variance of terms
- Paremiology: how are proverbs and idiomatic expressions rendered in other languages?
- Language learning: parallel reading, translation tests

# Source Code

Thanks for listening!
Source code available on Github at
https://github.com/pavan245/bitext-aligner